# Random Thoughts on Usability, Understandability, Programmability

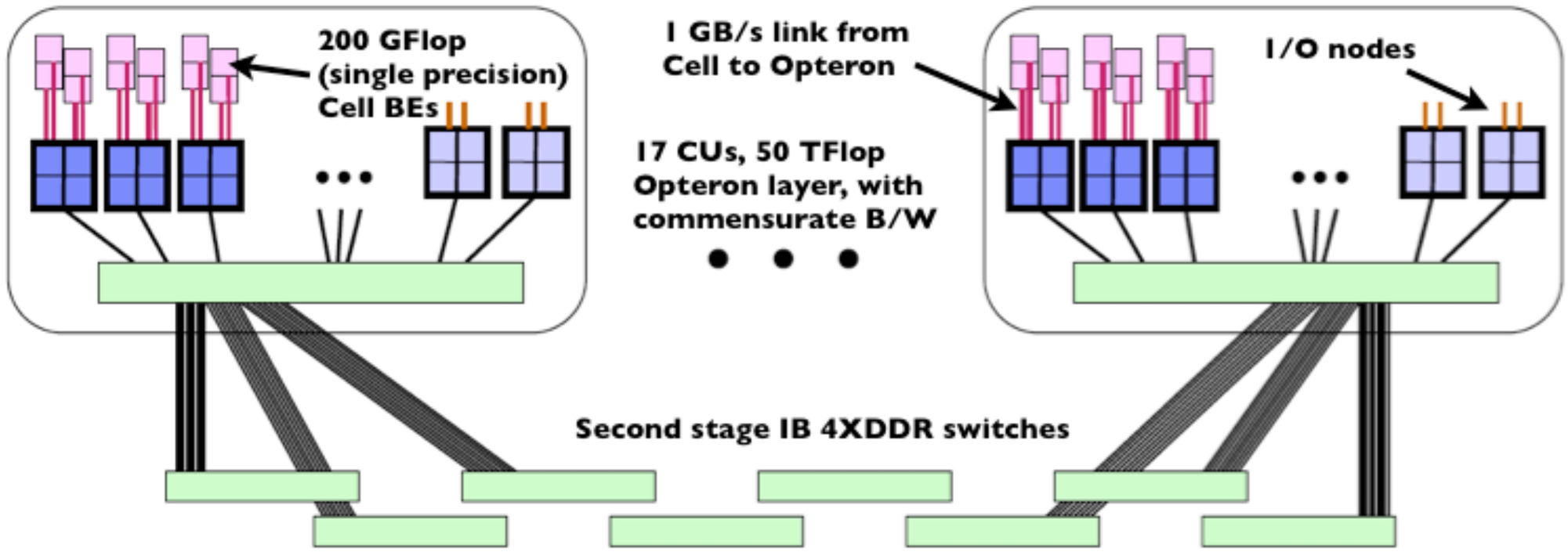
Salman Habib
HEP and MCS Divisions
Argonne National Laboratory

# Architectural Challenges are Real*

*in case there's any doubt in people's minds —

Andrew White Dec 7, 2007 + What if you had a petaflop/s

**Roadrunner:** Prototype for modern accelerated architectures, first to break the PFlops barrier




## Architectural 'Features'

- Complex heterogeneous nodes
- Simpler cores, lower memory/core, no real cache
- Skewed compute/communication balance
- Programming models?
- I/O? File systems?
- Effect on code longevity

HACC team meets Roadrunner

# Supercomputing: System Evolution

- **HPC imbalances (somewhat independent of heterogeneity)**
  - HPC systems were meant to be balanced under certain metrics — nominal scores of unity (1990's desiderata)
  - These metrics now range from ~0.1 to ~0.001 on the same system currently and are getting worse (out of balance systems)
  - RAM is expensive: memory bytes will not scale like compute flops, era of weak scaling (fixed relative problem size) has ended (at least for now)
- **Challenges**
  - Strong scaling regime (fixed absolute problem size) is much harder than weak scaling (since metric really is 'performance' and not 'scaling')
  - Machine models are complicated (multiple hierarchies of compute/ memory/network) — **heterogeneity!**
  - Applications action items: add more physics to use available compute, adds more complexity — may make the locality problem worse
  - Portability across architecture choices must be addressed (programming models, algorithmic choices, heterogeneity, trade-offs, etc.), "hard" vs. "soft" portability

# Co-Design vs. Code Design

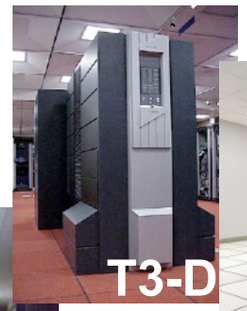
- **HPC 'Dreams/Myths'**
  - The magic compiler/programming model/language/ —
  - Co-Design (historically too perturbative, can this change?)
- **Dealing with Today's Reality (Defensive Code Design)**
  - Code teams must understand all levels of the system architecture, but not be enslaved by it (software cycles are long)!
  - Must have a good idea of the 'boundary conditions' (what may be available, what is doable, etc.)
  - 'Code Ports' is ultimately a false notion, need an architecture-aware code design philosophy and a helpful programming environment (rich/ interactive "decoration/pragmas?")
  - Special-purpose (domain-specific) hardware, some small(er) teams are looking at such options, hard to imagine this as a community solution
  - Need to start thinking out of the box — domain scientists and computer scientists and engineers must work together

# Handling the Transition —

- **Codes and Teams**
  - Most codes are written and maintained by small teams working near the limits of their capability (no free cycles)
  - Community codes, by definition, are associated with large inertia (not easy to change standards, untangle lower-level pieces of code from higher-level organization, find the people required that have the expertise, etc.)
  - Lack of consistent programming model for “scale-up”: **major issue**
  - Role of higher level languages: Python, R, —
- **Use Cases and Timelines**
  - Transitions needs to be staged (not enough manpower to entirely rewrite code base)
  - Prediction: There will be no ready made solutions, sad but likely true
  - Major problems for data-intensive applications (not used to designing for performance or caring about it)
  - Software cycles vs. hardware cycles (hard to predict precisely)
  - HPC systems are fragile, and this is getting to be more of a problem

# Exascale Review Recap



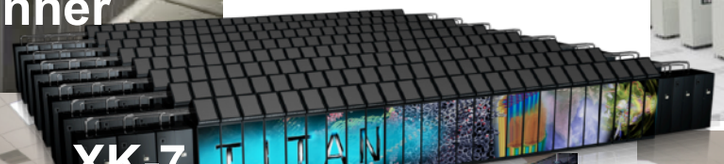






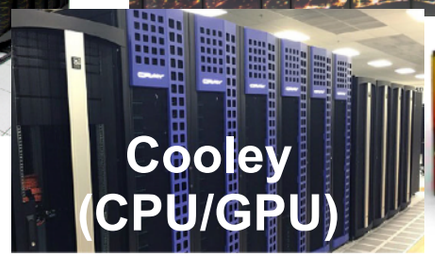










- **System Evolution**
  - Usability has gotten worse over time
  - Programmability has gotten worse
  - Understandability has gotten worse
  - **How much worse can it get?**
  - Exascale requirements review provided useful insights

- **General consensus from computationally experienced 'power' teams**
  - Common tools across architectures, efficient parallel primitives (can be opaque)
  - Should possess flexible data layouts to better handle memory hierarchies
  - Potentially replacable with hand-tuned code